A MACHINE LEARNING ALGORITHM IN AUTOMATED TEXT CATEGORIZATION OF LEGACY ARCHIVES

[1]Dali Wang, [2]Ying Bai and [1]David Hamblin

[1]Christopher Newport University, Newport News, VA, USA
[2]Johnson C. Smith University, Charlotte, NC, USA

ABSTRACT

The goal of this research is to develop an algorithm to automatically retrieve critical information from raw data files in NASA's airborne measurement data archive. The product has to meet specific metrics in term of accuracy, robustness and usability, as the initial decision-tree based development has shown limited applicability due to its resource intensive characteristics. We have developed an innovative solution that is much less resource intensive while offering comparable performance. As with many practical applications, the data available are noisy and correlated; and there is a wide range of features that are associated with the information to be retrieved. The proposed algorithm uses a decision tree to select features and determine their weights. A weighted Naive Bayes is used due to the presence of highly correlated inputs. The development has been successfully deployed in an industrial scale, and the results show that the development is well-balanced in term of performance and resource requirements

KEYWORDS

Machine Learning, Classification, Naïve Bayes, Decision Tree

1. BACKGROUND

Beginning in the 1980s, NASA (National Aeronautics and Space Administration) introduced tropospheric chemistry studies to gather information about the contents of the air. These studies involve a data gathering process through the use of airplanes mounted with on-board instruments, which gather multiple measurements over specific regions of the air. These measurements are complemented with other sources, such as satellites and ground stations. The data collection compiled, called missions or campaigns, are supported by the federal government with the goal of collecting data for scientific studies on climate and air quality issues. For the last two decades, data has been compiled into a single format, ICARTT (International Consortium for Atmospheric Research on Transport and Transformation) [1], which is designed to allow for sharing across the airborne community, with other agencies, universities, or any other group working with the data.

A typical ICARTT file contains, at the minimum, about 30 fields of metadata that describe the data gathered, the who/where/what aspects of the data gathering process, along with additional comments supplied by the principal investigator that is pertinent to the data. Over the past few years, the federal government has launched the Open Data Initiative, part of which aims at scaling

Dhinaharan Nagamalai et al. (Eds) : SAI, NCO, SOFT, ICAITA, CDKP, CMC, SIGNAL - 2019
pp. 01-08, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90701

up open data efforts in the climate sector. Accordingly, NASA has made it a priority to open up its data archive to the general public in order to promote their usage in scientific studies. As part of this initiative, NASA has introduced a centralized data archive service, which allows for others to quickly find certain information within the data archive by filtering data according to specific metadata components [2].

It is commonly believed that the most important information in an ICARTT data archive is the type of measurement taken by that experiment. Some of the examples include aerosol (such as aerosol particle absorption coefficient at blue wavelengths), trace gas (such as ozone mixing ratio), and cloud property (such as cloud surface area density). In order to uniquely define each measurement, NASA has adopted a naming system called "common names"; each common name is used for indexing a measurement with a uniquely-defined physical or chemical characteristic.

Unfortunately, the common name information is not directly presented in the ICARTT data file. Instead, each measurement is given a "variable name". Principal investigators decide each variable name in compilation of the ICARTT files, and along with the unique metadata it offers, the variable name is not easily searchable. Although they can be found in different missions, conducted by separate investigators, much of the data associated with a certain variable may be equivalent to others, save for the distinguishing metadata. For example, the gathering of formaldehyde (CH2O) in the troposphere is done in several missions across a number of years, all with differing variable names and through different sources. However, in a centralized data archive, all of these variables should be given the same common name to identify them as CH2O data. This more general common name allows a user to easily access similar data across missions by filtering based on this name.

The identification of a common name for each measurement variable was done by domain experts at NASA. The task is rather complicated as many fields in an ICARRT file contribute to the common name of a measurement: variable name, variable description, instrument used and measurement unit, etc. There are other information, although less intuitive, that may also contribute to the naming of each measurement. One such example is the principal investigator's (PI) name, because each person has his/her tendency in naming a variable. In addition, the data are noisy – missing fields and typos are common – as there has been no rigorous vetting process for metadata fields. As a result, there are only a few domain experts in NASA who can accurately identify a common name by studying the ICARRT file under consideration. Given the vast amount of data accumulated over the past 30 years, it is essential to automate the information retrieval process.

2. OVERVIEW OF SOLUTIONS

In its most common form, the common naming process is to choose a specific label (i.e., a common name) for a measurement variable (text) from a list of predefined labels, given the metadata associated with this variable. It belongs to the category of text categorization problems.

Textual similarity comparisons using fuzzy logic is a common approach to decide on matched and non-matched records [3]. It presents an opportunity for common name matching; however, given the variations in the type of metadata associated with each common name, and the large number of common names (300+), it becomes impractical to implement such an approach.

A forward-chaining based expert system, which belongs to a category of artificial intelligence algorithms, may provide another viable solution [4]. This approach was attempted by summarizing the rules used by domain experts and synthesizing a knowledge base. It turns out to not be very effective, as the rules are complex, non-inclusive, and sometimes contradictory. Also, whenever new missions are conducted, an expansion of rules in the inference engine is required - a tedious task.

Solutions to similar problems include Artificial Neural Networks (ANN) especially the perceptron-based network, Support Vector Machines (SVM), instance-based learning especially the k-Nearest Neighbor (k-NN) algorithm. Generally, SVMs and ANN tend to perform much better when dealing with multi-dimensional and continuous-valued features [5]. However, this problem only involves single dimensional and discrete features. The k-NN algorithm is very sensitive to irrelevant features and intolerant to noise. These make it unattractive for this problem, as not all features are relevant to every common name, and the information in general is very noisy due to errors and inconsistencies in ICARTT files.

Logic-based systems such as decision trees tend to perform better when dealing with discrete/categorical features. One of the most useful characteristics of decision trees is their comprehensibility. Typical decision trees, such as C4.5, are unreliable for classifying the airborne data. Preliminary testing with the algorithms using missions' data led to low overall accuracy values, due to their preference for binary and discrete features. Instead, a separate type of decision tree is considered. Classification and regression trees (CART) have been used in the statistics community for over thirty years [6], and are well-proven methods for creating predictive models in the form of a "tree", where branches are features and the end of a branch - the leaf - is the chosen class. Classification trees in particular have provided good results for datasets with categorical classes and features [7], while regression trees deal with continuous values unlike the atmospheric dataset. Classification trees are constructed through the partitioning of features based upon providing good splits, utilizing some metric to determine the "best" split. As they have the capability to work with datasets filled with categorical information and seeking a categorical outcome, classification trees align with the common naming process. Favorable preliminary results for implementation of a classification tree model lead to further investigation into utilizing this algorithm.

We first implemented CART, the Boosted CART and Bagged CART algorithms as solution for the common naming problem [8]. The algorithms provide the accuracy as well as robustness that we have aimed at. However, there are two major roadblocks in the deployment of the implementation. The first one is the execution time of the algorithms: the running time ranges from hours (for CART) to over 10 hours (for Boosted and Bagged version) using normal office computing power. This might be acceptable if the training is a one-time event. However, the training is necessary whenever a new mission is added to the data archive, because the newer missions may require the insertion of new common names.

The time complexity of CART-based algorithm is shown in Table I, where the complexity is a function of D (data space), L (label or class space), F (feature space), V_f value space for each feature, and M (number of trees).

Table 1. Time complexity of algorithms

Algorithm	CART	Boosted CART	Bagged CART
Complexity	$O(\|D\|\cdot\|L\|\cdot\|F\|\cdot 2^{\|V_t\|})$	$O(M\|D\|\cdot\|L\|\cdot\|F\|\cdot 2^{\|V_t\|})$	$O(M\|D\|\cdot\|L\|\cdot\|F\|\cdot 2^{\|V_t\|})$

As shown by the table's entries, CART is limited by the speed at which the algorithm constructs its classification tree, which can be bogged down by the introduction of large categorical features - like principal investigator (PI) or instrument - with their many choices. As it recursively partitions each feature to create a split, CART has to check for $2_n - 1$ values, where n is equal to the number of possible values for a feature. The running time will get worse in the future when more missions are included, because the possible values of a feature (such as the list of PIs and instruments) can only grow over time.

The second drawback of the CART-based algorithms is their memory-intensive nature. The classifiers are typically usable for all of the datasets, save for the enhanced (boosted and bagged) CART algorithms on the largest dataset - aerosol. Neither bagging nor boosting is able to execute for the aerosol dataset because of the algorithm's limitations. Due to the nature of the algorithms, including the large number of predictors and samples, the training models built exceed the memory limitations of 64-bit R. While performing well for the other datasets with less samples and predictors, the inability to produce results for aerosol shows the limited usability of enhanced CART methods.

The limitations of the CART-based algorithms make them less desirable for our application. We therefore were tasked with exploring other algorithms that provide the accuracy and robustness needed, while at the same time, demonstrate time and memory efficiency.

3. PROPOSED SOLUTION

The statistical-based Bayesian approach is investigated in this research. The major advantage of the Naive Bayes classifier is its short computational time, small storage space requirement, and robustness to noise and missing values - all are highly desirable for this application. The time complexity of the algorithm is $O(|D||L|)$ (D: training data space, L: label space). In addition, the algorithm is very transparent, as the output is easily grasped by users through a probabilistic explanation [5][9].

The Naive Bayes algorithm assumes all features are independent. Its accuracy will be affected with the presence of feature correlation. In the common naming process, some features are highly correlated. For example, a variable and the instrument used to measure the variable are closely related. We address the issues with several enhanced features to the simple Naive Bayes algorithm, which will be illustrated in the next few sections.

3.1 WEIGHTED NAIVE BAYES

The Naive Bayes algorithm is derived from Bayes' theorem of describing the probability of a label (or class), based on prior knowledge, formulated as:

$$P(cn|f_1, f_2, \cdots, f_n) = \frac{P(cn)\prod_{i=1}^{n} P(f_i|cn)}{\prod_{i=1}^{n} P(f_i)}$$

where cn is a common name and f_i is a feature.

The first assumption of Naive Bayes is that all features are independent. The second assumption, which is associated with feature independence, is that all of the features associated with a label are equally important. These assumptions, when violated, negatively affect its accuracy and make Naive Bayes less tolerant to redundant features. Numerous methods have been proposed in order to improve the performance of the algorithm by alleviating feature independence. A comprehensive study by Wu [10] on the entire 36 standard UCI data sets have shown that the weighted Naive Bayes provides higher classification accuracy than simple Naive Bayes.

Weighting Naive Bayes is accomplished with a single addition to the probability of features associated with common names:

$$P(cn|f_1, f_2, \cdots, f_n) = P(cn)\prod_{i=1}^{n}\left(\frac{P(f_i|cn_k)^{\omega_i}}{P(f_i)}\right)$$

For the purposes of common name matching, the probabilities are converted to logarithmic from linear for calculations; and the goal is to find the label x (common name) with the highest probability:

$$x = argmax[\, logP(cn_k) + \sum_{i=1}^{n}(logP(f_i|cn_k)\cdot\omega_i - logP(f_i))]for\ k \in \{1, \cdots, K\}$$

where $\{cn_k\}$ is the set of all possible common names.

3.2 WEIGHT CALCULATION

We use a separate classifier to determine the importance of features. In this case, an original CART decision tree [7] is utilized. In summary, a certain number of randomly sampled (with replacement) samples are assembled on differing proportions of the training set, and several classifiers are constructed based upon these samples for a given number of iterations. When building a classifier, not all features from the training set are used and some are considered more important than others, placed at higher depths within the tree. For each iteration of the classifier, feature weights are calculated based upon the depth (d) at which they are found. The weight for feature i from the k-th iteration is calculated as:

$$\omega_i^k = \frac{1}{\sqrt{d_k}}$$

The process is highlighted in Figure 1. After all iterations, the average of each feature's weight throughout the process is taken as the final weight.

An example of such a tree constructed for J variables is shown in Figure 2.

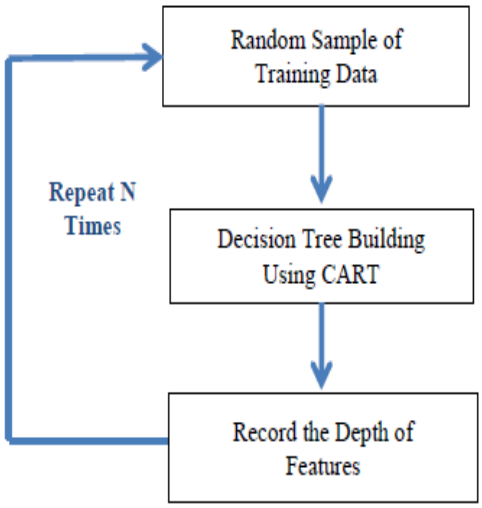

Figure 1 CART Algorithm for Weight Calculation

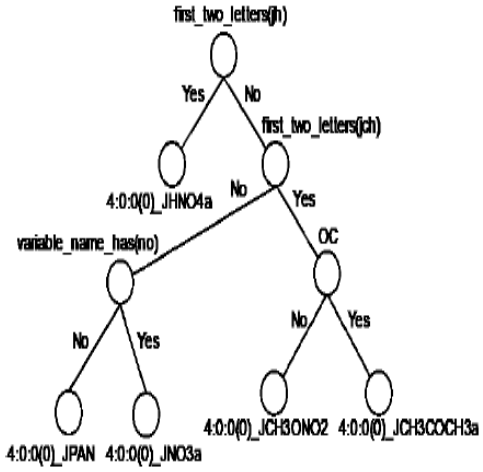

Figure 2 A Example Decision Tree

4. ALGORITHM DEVELOPMENT AND RESULTS

The weighted Naive Bayes algorithm is developed using Python. The software release also comes with an interactive user interface that allows user interaction before and during the algorithm's execution. Two modes of operation are implemented: batch mode, which produces all common name predictions at once (and outputs the results into a file); and interactive mode, which spits out one prediction at a time, and prompts the user for further operation. Due to its short execution time, users mostly start from a batch mode in order to gain a sense of overall results, which is followed by the interactive mode for more control.

As a statistical learning algorithm, weighted Naive Bayes is able to provide multiple predictions with a confidence level for each prediction. This feature becomes very desirable as the user can decide whether a domain expert's input is necessary based on the predictor's confidence level. The development has another very desirable feature: it allows for retraining of the classifier on-the-fly upon the request from a user. In the case that the algorithm misclassifies a measurement, the input from a domain expert is memorized by the algorithm, and classifier retraining can be started immediately. This ensures that future use of the algorithm will be based on an updated knowledge base, or the same misclassification is unlikely to happen again.

The training set used for this research includes a combination of eight different missions over a period of 15 years. A summary of the data set, listed in alphabetical order, is given in Table II. These are the data sets undergoing the data ingestion process during this research work. The results are determined using k-fold cross validation, breaking all mission data into k subsets, and testing is done through leave-one-out-cross-validation. This is to emulate the real world application of the algorithm. The performance of our software release using weighted Naive Bayes is summarized in Table III. In comparison, we also list the performance of the unweighted

Naive Bayes algorithm. Two aggregate measurements are also shown: the average accuracy of eight missions, and the weighted average accuracy by the number of variables.

Table 2. Dataset by missions

Mission	Year	Number of Files	Number of Variables	Number of Unique Common Names
ARCPAC	2008	27	91	41
CalNex	2010	31	266	193
DISCOVERAQ California	2013	29	622	100
DISCOVERAQ Maryland	2011	32	428	78
DISCOVERAQ Colorado	2014	23	371	101
DISCOVERAQ Texas	2013	19	224	92
NEAQS - ITCT2004	2004	22	799	160
TexasAQS	2000	30	261	200

Table 3. Accuracy of algorithms

Mission	Unweighted NB	Weighted NB
ARCPAC	0.64	0.69
CalNex	0.85	0.89
DISCOVERAQ California	0.95	0.97
DISCOVERAQ Maryland	0.96	0.97
DISCOVERAQ Colorado	0.95	0.96
DISCOVERAQ Texas	0.91	0.92
NEAQS-ITCT2004	0.81	0.84
TexasAQS	0.75	0.79
Average by Missions	0.85	0.89
Average by Variables	0.87	0.90

The results show that the weighted Naive Bayes outperforms the unweighted version consistently in very mission we tested. The overall performance, measured either by the average of all the missions or the weighted average of variables, also demonstrates the improvement by the weighted approach. Furthermore, the two algorithms are very similar in term their computational and memory complexity. The difference in execution time between the two algorithms is not even noticeable in our testing.

5. CONCLUSION

The contribution of this work is to develop a hybrid machine learning algorithm in order to achieve a good performance with reasonable computational complexity. The proposed approach harnesses the strength of both algorithms: CART ranks features automatically and is robust on noisy data, whereas Naive Bayes is very computationally efficient. The CART algorithms

developed prior to this research were very memory demanding and computational intensive. In some large data sets, it exceeded memory limitation of the running environment, and it took hours to run on a well-equipped office computer. Therefore, the objectives set for us were to achieve a specified performance metrics (which we did) while making the implementation practical for day-to-day operations.

Overall, an assessment of the weighted Naive Bayes algorithm with recent NASA data shows that the solution delivers robust results. It exceeds the performance expectation in the presence of inconsistencies and inaccuracies among measurement data. The software developed shows satisfactory results, runs quickly and needs very little memory space. Although the process involves decision-tree based training using CART, it is fundamentally different from the process that uses a CART algorithm as the sole solution for the problem. Since the ICARTT format remains the same, the feature selection will not change from mission to mission. Therefore, the CART algorithm only needs to be run once in order to obtain features and their weights, which in turn are used by the weighted Naive Bayes in all future applications. With its low storage requirement (several megabytes versus gigabytes for CART) and short execution time (a few minutes versus several hours for CART on a typical office computer), the weighted Naive Bayes presents a practical solution to the common name identification problem.

REFERENCES

[1]. A. Aknan, G. Chen, J. Crawford, and E. Williams, ICARTT File Format Standards V1.1, International Consortium for Atmospheric Research on Transport and Transformation, 2013

[2]. Matthew Rutherford, Nathan Typanski, Dali Wang and Gao Chen, "Processing of ICARTT data files using fuzzy matching and parser combinators", 2014 International Conference on Artificial Intelligence (ICAI'14), pp. 217-220, Las Vegas, July 2014

[3]. Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti and Rajeev Motwani, "Robust and Efficient Fuzzy Match for Online Data Cleaning", Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, June 2003, San Diego, CA, Pages 313 – 324.

[4]. B. G. Buchanan and E. H. Shortliffe, "Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project", Boston: MA, 1984.

[5]. S.B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques", Informatica, Vol. 31, 2007, Pages 249-268

[6]. L. Breiman, J. H. Friedman, R. Olshen, and C. J. Stone, Classification and Regression Trees (Wadsworth, Belmont, CA, 1984).

[7]. W. Y. Loh, "Classification and regression trees," WIREs Data Mining Knowl Discovery (2011).

[8]. David Hamblin, Dali Wang, Gao Chen and Ying Bai, "Investigation of Machine Learning Algorithms for the Classification of Atmospheric Measurements", Proceedings of the 2017 International Conference on Artificial Intelligence, page 115-119, Las Vegas, 2017

[9]. Tjen-Sien Lim, Wei-Yin Loh, Yu-Shan Shih, "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms", *Machine Learning Vol.* 40, 2000, Pages 203–228.

[10]. Jia Wu and Zhihua Cai, "Attribute Weighting via Differential Evolution Algorithm for Attribute Weighted Naive Bayes", Journal of Computational Information Systems, Vol. 7, Issue 5, 2011, Pages 1672-1679